

TASK1

Aim:- To determine the two halves of a given even-length string are alike by checking if both halves contain an equal number of vowels.

Algorithm:-

1. Start
2. Read the input string s.
3. Find the midpoint of the string using $\text{mid} = \text{length} / 2$.
4. Initialize two counters countA and countB to zero.
5. Define a string containing all vowels (both uppercase and lowercase).
6. Traverse from index **0** to mid - 1:
 7. If the character in the first half is a vowel, increment countA.
 8. If the corresponding character in the second half is a vowel, increment countB.
 9. Compare countA and countB.
10. If both counts are equal, return true; otherwise, return false.
11. stop

Program:-

```
class Solution {  
    public boolean halvesAreAlike(String s) {  
        int n = s.length(), count = 0;  
        String vowels = "aeiouAEIOU";  
        for (int i = 0; i < n / 2; i++)  
            if (vowels.indexOf(s.charAt(i)) >= 0) count++;  
        for (int i = n / 2; i < n; i++)  
            if (vowels.indexOf(s.charAt(i)) >= 0) count--;  
        return count == 0;  
    }  
}
```

Output:-

<input checked="" type="checkbox"/> Case 1	<input checked="" type="checkbox"/> Case 2
Input	Input
s = "book"	s = "textbook"
Output	Output
true	false
Expected	Expected
true	false

 Contribute a testcase

Result:- Thus, To determine the two halves of a given even-length string are alike by checking if both halves contain an equal number of vowels is successfully executed.

Task2:

Aim:- To check whether a given string is a **lapindrome**. A string is said to be a lapindrome if, after splitting it into two halves (ignoring the middle character if the length is odd), both halves contain the same characters with the same frequency.

Algorithm:-

1. Start.
2. Read the number of test cases T.
3. For each test case:
 - o Read the string S.
 - o Find the length n of the string.
 - o Calculate the middle index $mid = n / 2$.
4. Create two frequency arrays of size 26 (for lowercase English letters).
5. Count the frequency of characters in the first half of the string.
6. Count the frequency of characters in the second half of the string:
 - o If the string length is odd, skip the middle character.
7. Compare both frequency arrays:
 - o If all frequencies match, the string is a lapindrome.
 - o Otherwise, it is not a lapindrome.
8. Print "YES" if the string is a lapindrome, else print "NO".
9. Stop

Program :-

```
import java.util.*;  
  
import java.lang.*;  
  
import java.io.*;  
  
class Codechef {
```

```
public static void main (String[] args) throws java.lang.Exception {  
    Scanner sc = new Scanner(System.in);  
  
    int T = sc.nextInt();  
  
    while (T-- > 0) {  
  
        String s = sc.next();  
  
        int n = s.length();  
  
        int[] freq1 = new int[26];  
  
        int[] freq2 = new int[26];  
  
        int mid = n / 2;  
  
  
  
        for (int i = 0; i < mid; i++) {  
  
            freq1[s.charAt(i) - 'a']++;  
  
        }  
  
        for (int i = (n % 2 == 0 ? mid : mid + 1); i < n; i++) {  
  
            freq2[s.charAt(i) - 'a']++;  
  
        }  
  
        boolean isLapindrome = true;  
  
        for (int i = 0; i < 26; i++) {  
  
            if (freq1[i] != freq2[i]) {  
  
                isLapindrome = false;  
  
                break;  
            }  
        }  
    }  
}
```

```

        }

    }

    if (isLapindrome)
        System.out.println("YES");

    else

        System.out.println("NO");

    }

    sc.close();

}

}

```

Output:-

The screenshot shows a terminal window with the following details:

- Status :** Successfully executed
- Time:** 0.0700 secs
- Memory:** 42.136 Mb
- Sample Input:**

```

6
gaga
abcde
rotor
xyzxy
abbaab
ababc

```
- Your Output:**

```

YES
NO
YES
YES
NO
NO

```

Result:- Thus, The program successfully identifies whether each given string is a lapindrome or not by comparing the character frequencies of its two halves and displays “YES” for lapindrome strings and “NO” for non-lapindrome strings are successfully executed.